AD-A056 761 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 5/1
A NETWORK AUGMENTING PATH BASIS ALGORITHM FOR TRANSSHIPMENT PRO--ETC(U)
MAR 78 R BARR, J ELAM, F GLOVER, D KLINGMAN N00014-75-C-0569
UNCLASSIFIED CCS-272 NL

| OF |
AD
A056761

END
DATE
FILMED
9-78
DDC

AD A056761

AD No.
DDC FILE COPY

LEVEL II

12

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

DDC
RECEIVED
JUL 28 1978
F

This document has been approved for public release and sale; its distribution is unlimited.

78 07 24 006

AD A056761

LEVEL II

AD No. ___ DDC FILE COPY

(11)

(14) CCS-272

A NETWORK AUGMENTING PATH BASIS ALGORITHM FOR TRANSSHIPMENT PROBLEMS,

(9) Research rept.,

by

(10) Richard/Barr*
Joyce/Elam**
Fred/Glover***
Darwin/Klingman****

DDC RECEIVED JUL 28 1978 F

August 1977
Revised Mar[illegible]78

(11)

(12) 35p.

* Visiting Research Fellow, Department of the Treasury, Office of Tax Analysis, Washington, D.C. 20220 (on leave from Southern Methodist University, Dallas, Texas)

** Assistant Professor of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104

*** Professor of Management Science, University of Colorado, Boulder, CO 80302

**** Professor of Operations Research and Computer Sciences, BEB 608, The University of Texas, Austin, TX 78712

(15) This research was partly supported by ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute and by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

N00014-75-C-0616

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 203E
The University of Texas
Austin, Texas 78712
(512) 471-1821

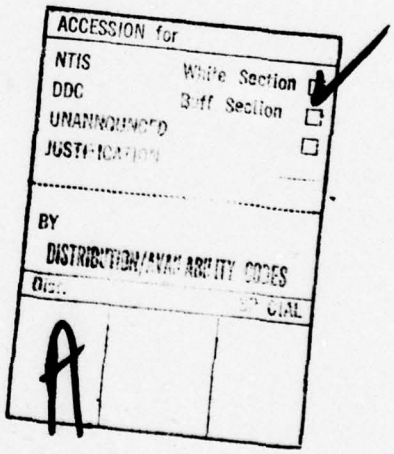
This document has been approved for public release and sale; its distribution is unlimited.

406 197

78 07 24 006 LB

ABSTRACT

The purpose of this paper is to present a new simplex algorithm for solving capacitated transshipment network problems which both circumvents and exploits the pervasive degeneracy in such problems. This generalized alternating path algorithm is based on the characterization of a special subset of the bases that are capable of leading to an optimal solution. With consideration restricted to these bases, fewer alternative representations of a given extreme point are inspected. The impact on the number of degenerate pivots and problem solution times is demonstrated by computational testing and comparison with other approaches.



1. INTRODUCTION

In this paper, a new primal simplex algorithm for solving capacitated transshipment problems is presented. This method exploits the combinatorial possibilities available in degeneracy to obtain significant theoretical and computational advances. An important theoretical aspect of the algorithm is that it is finitely convergent. Cunningham [6] derived the same theoretical results in his development of a similar algorithm. His excellent paper provides the first breakthrough for handling degeneracy in capacitated transshipment problems following the introduction of perturbation and lexicographical ordering techniques for ensuring finiteness in primal simplex algorithms.

Degeneracy, however, not only creates a theoretical problem, but also constitutes a major computational issue. In fact, computational studies have shown the number of degenerate pivots executed when solving large capacitated transshipment problems to range upward from 70%. The new primal algorithm was independently discovered as a result of devising computational schemes for avoiding and efficiently performing degenerate pivots. The motivation for the algorithm came from *practical* experience, and the algorithmic design came from studying the graph characteristics of degenerate bases.

This presentation extends the work of Cunningham to include insights into the computer implementation of the new algorithm. Labeling techniques are provided for implementing the algorithm and computational results based on one such implementation are presented. These results demonstrate the merits

of the techniques, with particular reference to computational efficiencies achieved in solution time and storage requirements. This algorithm may also be viewed as an extension of those of [3, 4, 5] for solving assignment, semi-assignment, and transportation problems. Thus this presentation provides a bridge between earlier work in the literature and recent advances in network methodology.

Degenerate basic variables and pivots, which are particularly prevalent in large network problems, result from the large number of possible alternative basis representations of the extreme points. In order to reduce the number of such representations considered (hence the number of possible degenerate pivots) and to provide a mechanism whereby degenerate pivots can be exploited, the new primal simplex algorithm narrows consideration to a special subset of bases: those with a network augmenting path (NAP) structure. The NAP structure is a generalization of the traditional alternating path structures used in graph theory (e.g., in the context of matching problems [7, 8]) and also of the generalized alternating path structures introduced in [5]. If a network problem has an optimal solution, then an optimal solution can be found by considering only bases that have the NAP structure.

The algorithm, hereafter referred to as the network augmenting path (NAP) algorithm, possesses the following computational properties:

(1) No tests are required to ensure that only bases with the NAP structure are examined.

(2) The algorithm is finitely convergent without reliance on lexicographic or perturbation techniques.

(3) In certain cases, the type of basis exchange (degenerate or non-degenerate) may be recognized prior to finding the entire representation of an incoming arc.

The computational results of Section 5 indicate that these properties provide a more efficient algorithm than other special purpose simplex network algorithms [1, 2, 9, 12].

2. BACKGROUND

The *capacitated minimum cost flow network* or *capacitated transshipment* problem can be stated as follows:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$\text{Subject to:} \quad \sum_{i\in I_k} x_{ik} - \sum_{j\in O_k} x_{kj} = b_k, \ k\in N \tag{2}$$

$$0 \le x_{ij} \le u_{ij} \text{ for } (i,j)\in A \tag{3}$$

where $I_k = \{i\in N:(i,k)\in A\}$ and $O_k = \{j\in N:(k,j)\in A\}$. In standard terminology A is the set of arcs (i,j) of the network G(N,A) and N is the set of nodes. The constant b_k represents the "requirement" at node k, which is frequently referred to as the supply if $b_k < 0$ and as the demand if $b_k > 0$. Associated with each node $k\in N$ is a dual variable π_k called its *node potential*. An arc (i,j) is directed *from* node i (the tail node) to node j (the head node). An arc (i,j) is also said to be *out-directed* from node i and *in-directed* to node j. Thus, in particular, I_k is the set of tail nodes of arcs that are in-directed to node k, and O_k is the set of head nodes of arcs that are out-directed from node k.

The flow, cost, and upper bound of arc (i,j) are represented, respectively, by x_{ij}, c_{ij}, and u_{ij}. In the terminology of simplex (extreme point) solution algorithms for networks, the reduced cost of arc $(i,j)\in A$ is $\bar{c}_{ij} = c_{ij} + \pi_i - \pi_j$. The objective is to determine a set of arc flows which satisfies the node requirements and capacity restrictions at minimum total cost.

Graphical Structure of Network Bases

A bounded variable simplex basis for a network flow problem corresponds to a spanning tree with $|N| - 1$ arcs (where $|N|$ denotes the cardinality of the set N). An arc is called *basic* if it is contained among those arcs in the basis tree and is called *nonbasic* otherwise. Each nonbasic arc has a flow equal to 0 or to its upper bound.

Once the flows on the nonbasic arcs have been set, the flows on the basic arcs are uniquely assigned so that equation (2) is satisfied. If equation (3) is also satisfied, this assignment of flows is a *basic feasible solution*. For each basis, node potentials are assigned values that satisfy *complementary slackness*; that is, these node potential values are determined so that the reduced cost for each basic arc is zero. Figure 1 illustrates a spanning tree basis for a network problem. The arcs are drawn according to the direction they receive in the network. The number beside each arc indicates the flow on this

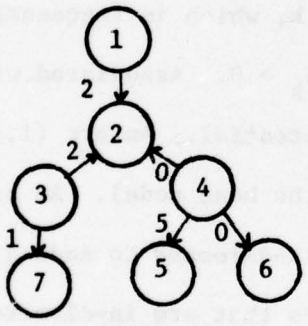


Figure 1
Basis Graph

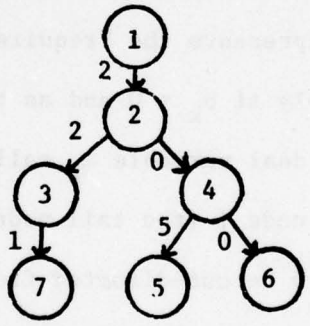


Figure 2
Rooted Tree Basis Graph

NODE	PRED	CONF	FLOW
1	None	--	--
2	1	1	2
3	2	0	2
4	2	0	0
5	4	1	5
6	4	1	0
7	3	1	1

Table I
Representation of Rooted Tree Basis

arc imparted by the basic solution.

Rooting the spanning tree basis

The most efficient procedures for solving network flow problems [1, 9, 10, 12] are based on storing the basis as a *rooted tree*. A rooted tree is often visualized as having its root node at the top with all other nodes in the tree lying on directed paths leading downward from the root. Since such a tree more nearly resembles a geneological tree than a botanical tree, an *ancestor* of x is defined as any node on the unique path from the root to x (including x itself), and a *descendant* of x is defined to be any node for which x is an ancestor. (An ancestor or descendant of x other than x itself will be called a strict ancestor or descendant.) The *predecessor* of x is its first (closest) strict ancestor, and a *successor* of x is any node for which x is its predecessor. The root node has no precedessor.

When a basis is stored as a rooted tree, each basic arc is given an orientation which has nothing to do with its actual direction in the network. To distinguish an arc's direction in the network from its orientation in the basis, the basic arc (x,y) is said to be *conformably oriented* (or just *conformable*) if its orientation in the basis tree agrees with its actual direction in the network (i.e., x is the predecessor of y), and is said to be *nonconformably oriented* (or just *nonconformable*) otherwise.

Figure 2 illustrates the rooted tree representation of the basis shown in Figure 1. The information required to represent this rooted tree can be efficiently organized as shown in Table I. The arrays NODE and PRED contain the nodes and the predecessors of the nodes, respectively. Associated with each node (except for the root node) is a 1 or 0 entry in the CONF array which indicates respectively, the conformability or nonconformability of the basic arc that connects

the node to its predecessor. An entry in the FLOW array identifies the flow in the associated arc. (Note that the array NODE would not be stored in any implementation.)

Summary of the Network Simplex Method

The NAP algorithm is based upon a graphical interpretation of the simplex method as applied to network problems. For completeness, the steps of this simplex method specialization are summarized.

Step 1: Determine Node Potentials to Price Out the Basis.

Assume that a feasible basis (possibly containing artificial arcs) has been determined and stored as a rooted tree. It is then necessary to determine node potentials π_k for each node k so that the reduced cost equals 0 for each basic arc (i,j). Due to the redundancy in the defining equations of the network problem, one node potential can be set arbitrarily. Customarily, the node potential of the root is set to zero. The remaining node potentials can be determined in a cascading fashion by moving down the tree and identifying the potential for node j using the potential previously set for its predecessor, node i, and the equation

$$c_{ij} + \pi_i - \pi_j = 0$$

if the basic arc connecting nodes i and j is conformable (i.e., arc (i,j)),

$$c_{ji} - \pi_i + \pi_j = 0$$

if the basic arc connecting nodes i and j is nonconformable (i.e., arc (j,i)).

Step 2: Identify the outgoing and incoming arcs.

The fundamental basis exchange step of the simplex method selects an incoming arc and an outgoing arc from the set of nonbasic and basic arcs, respectively.

(a) The incoming arc: A nonbasic arc which is profitable [i.e., has zero flow and a negative reduced cost or has saturating (upper bound) flow and a positive reduced cost] is selected to enter the basis. If no such arc exists, the algorithm terminates. In the latter case, the solution is feasible and the current arc flows are optimal if no artificial arcs with positive flow exist; otherwise, the problem is infeasible.

(b) The outgoing arc: The arc to leave the basis is determined by tracing the unique basis path which connects the two nodes of the incoming arc. This *basis equivalent path* can be determined by tracing the predecessors of the two nodes to their point of intersection. An attempted change of the flow of the incoming arc in its profitable direction (away from the bound it currently equals) causes a change in the flows of the arcs contained in this basis equivalent path. In order to maintain primal feasibility, the outgoing arc must always be an arc in this path whose flow goes to zero or its upper bound ahead of (or at least as soon as) any others as a result of a flow change in the incoming arc. Such an arc is called a *blocking arc*. (This arc can, of course, be the same as the incoming arc for the capacitated problem.)

Step 3: Execute the basis exchange.

The outgoing and incoming arcs swap their basic/nonbasic statuses to become nonbasic and basic, respectively, whereupon a full iteration is completed and the method returns to Step 1 with a new primal feasible basis.

In the following results, the convention will be followed that the incoming arc be designated by (p,q) (directed from node p to node q in the network) if it has 0 flow before the pivot step, and that it be designated (q,p) (directed from node q to node p in the network) if it has saturating flow before the pivot step. Node z will designate the first common node on the predecessor paths from node p and node q to the root. Thus, the z-p path and the z-q path in the basis tree may

be referred to without ambiguity, where either (but not both) of these paths may contain no arcs--i.e., possibly z = p or z = q.

Basic Flow Changes

To characterize more precisely the flow changes which occur as a result of a basis exchange step, the intuitive notion of increasing (or decreasing) flow across a path from a descendant to an ancestor will be used.

Remark 1: A flow increase (decrease) across a path in a rooted tree basis from a descendant to ancestor creates a decrease (increase) in the flow of conformable arcs and an increase (decrease) in the flow of nonconformable arcs.

Proof: A flow change of Δ across a path causes a change of Δ in the flow of arcs whose tree orientation coincides with their network direction and a change of $-\Delta$, otherwise. Since a path from descendant to ancestor traverses nonconformable arcs in accordance with their network direction and conformable arcs in opposition to their direction, the result follows immediately.

A fundamental result which characterizes the exact flow changes that occur as a result of a basis exchange step is now stated.

Remark 2: In a rooted basis tree, a profitable flow change for the incoming arc (an increase if the arc is (p,q) and a decrease if the arc is (q,p)) decreases (increases) the flow of all conformable arcs and increases (decreases) the flow of all nonconformable arcs on the z-q path (z-p path) by a corresponding amount.

Proof: A flow change in the incoming arc results in a flow change to each arc in the z-p path and the z-q path. In particular, a flow increase on arc (p,q) may be viewed as sending a flow increase from q to z (i.e., from a descendant to an ancestor) along the z-q path and from z to p (i.e., from an ancestor to a descen-

dant) along the z-p path. Since a flow increase in one direction on a path is algebraically equivalent to a flow decrease in the opposite direction, a flow increase from z to p is equivalent to a flow decrease from p to z (i.e., from a descendant to an ancestor). Correspondingly, a flow decrease in arc (q,p) may be viewed as sending a flow decrease from p to z along the z-p path and a flow increase from q to z along the z-q path. Combining these statements with Remark 1, Remark 2 is obtained.

The use of the foregoing observations to identify the outgoing arc in a basis exchange step is illustrated as follows. Assume that the starting basis is the one given in Figure 2 and the incoming arc is (5,7), which is currently at its lower bound. The basis equivalent path for arc (5,7) consists of the z-p path from node 2 to node 5 and the z-q path from node 2 to node 7. As flow is increased on arc (5,7), flow is increased on the path from node 7 to node 2 and decreased on the path from node 5 to node 2. The amount of flow change that can be accommodated on the z-q path is determined by starting at node 7 and tracing predecessor until node 2 is encountered. The flow on conformable arcs (arc (3,7)) is decreased and the flow on nonconformable arcs (arc (3,2)) is increased until the flow on one of these arcs reaches a lower or upper bound and cannot be changed further without violating this bound. The allowable flow change on the z-p path is similarly determined by starting at node 5 and tracing predecessors until node 2 is encountered, increasing flow on conformable arcs (arc (4,5)) and decreasing flow on nonconformable arcs (arc (4,2)) until further change would violate a lower or upper bound. The arcs that limit the flow change most restrictively qualify as blocking arcs. In this case, arc (4,2) is already at its lower bound and cannot absorb any flow decrease, and therefore qualifies as a blocking arc. When arc (5,7) is brought into the basis, arc (4,2) must be re-

moved (since there are no other blocking arcs). In addition, the basis exchange is degenerate because no flow change occurs.

Special Updating Considerations

After the incoming and outgoing arcs have been selected, the basis exchange is completed by determining the updated flows on the arcs in the basis equivalent path and updated node potentials for the new basis tree. Only a subset of the node potentials change during a basis exchange step and these potentials can be updated by simplified marginal calculations rather than determined from scratch by solving the reduced cost equations. It is in fact possible to elect to change only the potentials of the nodes in one of the two subtrees created by removing the outgoing arc. For convenience in the following discussion, it will be assumed that the subtree selected for the operation of updating node potentials is the one that does not contain the root node (thereby assuring that the root node and its node potential remain unchanged from iteration to iteration). Thus, for example, the node potentials of the basis tree in Figure 2, which are updated as a result of adding arc (5,7) and removing arc (4,2) from the basis, are those contained in the subtree rooted at node 4.

Any implementation of the simplex method specialized to a graphical framework, must provide the ability to perform two types of tree traces, or traversals: an *upward traversal* to find basis equivalent paths and to update flows, and a *downward traversal* to identify all nodes of the subtree whose potentials must be updated as a result of a basis exchange. The most efficient procedure for performing these traversals in terms of both computer memory requirements and solution speed is the augmented threaded index (ATI) method [11]. The ATI method uses a special list structure which consists of a predecessor function and a thread function. Together these two functions (node labels) make it possible to identify

efficiently all predecessors and successors of a given node in the basis tree.

Figure 3 illustrates the thread function for the basis tree shown in Figure 2. At the expense of another node length array of computer memory, further computational advantages can be achieved by augmenting the ATI method with the distance function [14] or the cardinality function [14]. (See reference [2].) The distance function for a given node indicates the number of arcs on its predecessor path to the root. The cardinality function indicates the number of successors of a given node (hence one less than the number of nodes in the subtree rooted at the given node).

These labels can be integrated into the implementation of the new algorithm of this paper, as will be shown, to minimize the effort spent in performing degenerate pivots. Table II illustrates the predecessor, thread, distance, and cardinality function for each node in the basis tree shown in Figure 2. With this background, the fundamental definitions and relationships underlying the NAP algorithm are developed.

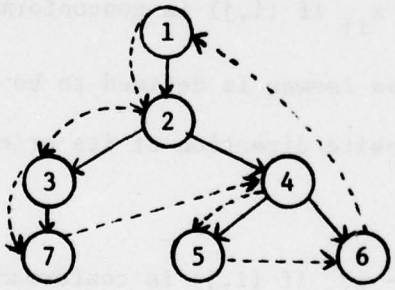


Figure 3

Thread Function

NODE	PRED	THREAD	DISTANCE	CARDINALITY
1	None	2	0	6
2	1	3	1	5
3	2	7	2	1
4	2	5	2	2
5	4	6	3	0
6	4	1	3	0
7	3	4	3	0

Table II
Label Functions for Rooted Tree Basis

3. NETWORK AUGMENTING PATH BASIS DEFINITION AND CONSTRUCTION

Fundamental Definitions

The *forward leeway* of an arc (i,j) is defined to be its net capacity to absorb a flow decrease in the direction that coincides with its orientation in the basis tree, or more precisely, to be

$$x_{ij} \text{ if } (i,j) \text{ is conformable or}$$

$$u_{ij} - x_{ij} \text{ if } (i,j) \text{ is nonconformable.}$$

Correspondingly, the *reverse leeway* is defined to be its net capacity to absorb a flow decrease in the opposite direction of its orientation in the basis tree, or more precisely, to be

$$u_{ij} - x_{ij} \text{ if } (i,j) \text{ is conformable or}$$

$$x_{ij} \text{ if } (i,j) \text{ is nonconformable.}$$

An arc which has both positive forward and reverse leeway is defined to have *double leeway*. Clearly, an increase in an arc's forward leeway is equivalent to a decrease in its reverse leeway, and conversely. Thus, since a flow decrease in one direction corresponds algebraically to a flow increase in the opposite direc-

tion, the reverse leeway of an arc may also be viewed as its net capacity to absorb a flow increase in the direction that coincides with its basis tree orientation.

NAP Basis Structure

A rooted basis tree for a capacitated network problem is a *NAP basis* if every arc (i,j) contained in the rooted basis tree has positive forward leeway. It follows directly from the definition of forward leeway that a rooted basis tree is a NAP basis if every zero flow arc in the basis is nonconformable and every saturated arc in the basis is conformable. A *network augmenting path (NAP)* is any path supplied with an orientation for which the foregoing definition applies. Thus, the path from an ancestor node to any descendant node in a NAP basis is a NAP.

A NAP generalizes the concept of an ordinary alternating path in graph theory, which consists of a sequence of edges whose odd members belong to one category and whose even members belong to a second category, by viewing all edges of the first category to be zero flow arcs whose path orientations conform to their arc directions, and all edges of the second category to be unit flow arcs whose path orientations do not conform to their arc direction. (No reference to capacitation is required.) This concept also extends the generalized alternating path (GAP) notion of [5], which similarly assumes that odd arcs are directed compatibly with their path orientations, and even arcs are directed otherwise, but which allows capacitation and flows other than zero or one.

NAP Basis Equivalence

Consider the rooted basis tree shown in Figure 1. This basis does not satisfy the definition of a NAP basis since a conformable arc has zero flow. In general,

it is clear that many bases for the network problem will not be NAP bases. Thus, the question arises whether a given basis can be given the NAP structure by a proper orientation of the basis tree and/or by interchanging the roles of forward leeway and reverse leeway (since it is entirely permissible to replace forward leeway in the definition of the NAP structure by reverse leeway). A basis will be called NAP equivalent if it can be endowed with the NAP structure in this manner. Every nondegenerate basis is trivially NAP equivalent, due to the fact that every arc has double leeway, and thus any orientation of the basis satisfies the NAP basis definition. The situation is more complex in the case of degeneracy and the restrictiveness of the structure with respect to degenerate bases is addressed in the following result.

Remark 3: A network basis is not NAP equivalent if any of the following conditions hold:

(1) there exists a node whose incident basic arcs include four out-directed arcs or four in-directed arcs such that two have zero flow and two have saturating flow;

(2) there exists a node whose incident basic arcs include two out-directed arcs and two in-directed arcs all with zero flow or all with saturating flow;

(3) there exist any two nodes such that the incident basic arcs of the first include two in-directed arcs and the incident basic arcs of the second include two out-directed arcs, and where the flow is zero or saturating (uniformly) on these four arcs;

(4) there exist any two nodes such that the incident basic arcs of each include two in-directed arcs or include two out-directed arcs, such that the flows are zero on the two arcs associated with one node and are saturating on the two arcs associated with the other node.

Proof: The result follows from the fact that in each case one pair of adjacent arcs implies that the standard definition of the NAP structure must be used, and the second pair of adjacent arcs implies that a definition which interchanges the rules of forward and reverse leeway must be used.

Generating a NAP Basis

In this section a procedure for constructing a NAP basis that contains the same flows as any given non-NAP basis is developed. This procedure consists of two stages as follows.

Stage 1:

Delete all arcs of the non-NAP basis tree whose flows equal their upper or lower bounds. The remaining portion of the network is a forest, some of whose trees may consist of isolated nodes. No orientations are assigned to these trees; that is, no node, isolated or otherwise, is assigned a predecessor index.

Stage 2:

This stage consists of one fewer iterations than the number of distinct trees inherited from Stage 1. At each iteration, two trees of the current forest are joined, reducing the total number of trees by one. To describe this stage, the *distinguished node* of an arc not in the forest is defined to be its tail node if the arc has zero flow and its head node if the arc has saturating flow. The operation of this stage then proceeds as follows. Select any two trees in the current forest and add any arc joining these two trees whose distinguished node has not been assigned a predecessor. (Subsequently it is shown how to specify such an arc easily, though it may not be an arc of the original basis.) Then make this distinguished node the root of all nodes in its subtree, and let the other node of the

arc just added be the predecessor of the distinguished node, thus making it the root of the extended subtree that includes the added arc. This assigns predecessor indexes to all nodes of the subtree, but not to the root of the extended subtree. Of course, the root of the extended subtree may already have a predecessor by an earlier operation. The process repeats until only one tree remains, whereupon any node of this tree without a predecessor is designated the root of the entire basis, and the construction is complete.

A special comment applies to the implementation of Stage 2. At each iteration (as will be shown), there exists an easily specified set of nodes without predecessors in each tree. Thus, for simplicity, one can legitimately add any arc between the two subtrees that links one such node to another. In the case of sparseness, where many arcs may not be present in the network, it may be convenient to introduce an artificial arc between two such nodes (e.g., to avoid a search among alternatives). This arc can then be assigned a "big M" cost and a zero flow (thus making its flow compatible with the flow values of the original basis, which likewise assigns a zero flow to the added arc once the arc is created).

The characteristics of the forest generated by the foregoing procedure are identified in the following lemma.

Lemma 1: At each iteration of Stage 2 of the NAP basis generating procedure, each tree of the current forest contains a *unique* unoriented subtree (possibly equal to the entire tree) which consists of one of the original trees inherited from Stage 1, and such that no node of this subtree has been assigned a predecessor index. The remainder of the tree consists of a collection of fully oriented subtrees, each rooted in the unique unoriented subtree by an arc that is one of the arcs added

during Stage 2 (though other arcs added during Stage 2 may not be among these). Finally, each of the oriented subtrees has the NAP basis structure, and no predecessor index is ever altered by a subsequent operation of joining two trees.

Proof: The result follows by induction. Clearly, the stated conditions hold at the beginning of Stage 2. Then, at each iteration, since the unique unoriented subtree is one of the original subtrees inherited from Stage 1 (which may just be an isolated node that has not been assigned a predecessor), it follows that every arc in this subtree has double leeway, and hence this subtree may compatibly receive any orientation whatsoever and still satisfy the requirements for the NAP basis structure. The condition that the distinguished node of the added arc has no predecessor avoids any conflict with previously assigned predecessors and thus preserves any previously existing NAP basis structure. The specific orientation of the added arc likewise preserves the structure. Finally, the fact that all of the oriented subtrees are rooted in the unique unoriented subtree guarantees that any node of the latter may be made the root for the entire tree without conflicting with existing orientations. This completes the proof.

The following main result is a consequence of this lemma:

Theorem 1: The final tree produced by the NAP basis generating procedure is a NAP basis, and assigns the same flows to all arcs as the original feasible network basis (where arcs of the original basis that are not in the NAP basis constitute nonbasic arcs for the latter, with flows equal to their upper or lower bounds).

The foregoing procedure for generating a NAP basis can, of course, also be superimposed on any method for generating a "starting" primal basis for a minimum cost flow network (which may require the introduction of artificial arcs with nonzero flows), and thus serve as a constructive means for obtaining a starting NAP basis.

4. IMPORTANT PROPERTIES OF NAP BASES AND THE NAP ALGORITHM

A NAP basis possesses three fundamental properties which distinguish it from bases normally given consideration by a primal simplex method. First, the structure of the basis allows one to characterize the portion of the basis graph in which degeneracy can occur. Secondly, for any choice of an incoming nonbasic arc in a basis exchange (pivot) step for a primal simplex method, there is a unique outgoing arc which can be selected to leave the basis that will maintain primal feasibility and preserve the NAP structure. Lastly, cycling through a series of degenerate NAP bases is prevented without the use of external techniques such as perturbation or lexicographical ordering. The remainder of this section provides the proofs for these properties and states the rules of the NAP algorithm.

Theorem 2: Degeneracy in a rooted basis tree that possesses the NAP structure, if it exists, must occur on the z-p path.

Proof: In a NAP basis, every arc has positive forward leeway. Thus, the existence of a blocking arc on the z-q path immediately implies nondegenercy from Remark 2. If the incoming arc is a blocking arc, the pivot is also nondegenerate. Thus, degeneracy, if it exists, must occur on the z-p path.

In stating the remaining properties of a NAP basis, it is convenient to refer to the augmented z-q path, designated as z-q-p path, from z to q and then to p via the arc (p,q) or (q,p), according to the flow on the arc before the pivot. The motivation for attaching the incoming arc to the z-q-p path is that this is precisely the one of the two possible attachments (the other yielding a z-p-q path) that preserves the NAP path structure for the current arc flows. For this purpose, an arc on the z-p or z-q-p path is said to be *higher* than another arc on this path if it lies closer to z and to be *lower* otherwise.

Theorem 3: The NAP basis structure is preserved by the pivot* if and only if the outgoing arc is selected to be:

- the highest blocking arc on the z-q-p path if any blocking arc occurs on this path, or

- the lowest blocking arc on the z-p path if no blocking arc occurs on the z-q-p path.

Proof: If a blocking arc occurs on the z-q-p path (hence the pivot is nondegenerate), then the increase in forward leeway on the z-p path (corresponding to the decrease in reverse leeway) simply reinforces the current NAP structure of this path, and hence the path can be left intact. On the other hand, the decrease in forward leeway on the z-q-p path destroys the NAP structure for any blocking arc on this path (whose forward leeway is reduced to zero). Dropping the highest blocking arc on this path, by selecting it to be the outgoing arc, thus preserves the NAP structure above the outgoing arc (and is clearly the only choice that will). Further, by the updating rule that maintains the current root, all arcs below this arc on the z-q-p path (if any exist, i.e., if the incoming arc itself is not the unique blocking arc on this path) are reoriented in the opposite sense so that p becomes the ancestor of these arcs. The fact that the pivot is nondegenerate, and thus that the reverse leeway for each arc on the z-q-p path is strictly increased (corresponding to the decrease in forward leeway) implies that

---

*It is assumed that the standard rule is used for efficiently reorienting the arcs of the basis after the pivot, without re-rooting (see e.g., [11]). This assumption is "non-restrictive" in that the customary process for re-rooting can readily be shown to destroy the NAP basis structure whenever a blocking arc exists on both the z-p and z-q paths (among other cases), and any other choice of a new root only entails similar hazards (unless the path to the old root is entirely free of arcs that have zero reverse leeway) but also requires other superfluous calculations in order to implement the algorithm.

each arc whose orientation has been reversed now has positive forward leeway, thus maintaining the NAP structure. Finally, if blocking arcs occur only in the z-p path, the applicable reversed orientation is again for the portion of the path below the dropped arc, and while the NAP structure is automatically preserved on the z-p path above *any* dropped arc (by the argument noted earlier), this structure will exist on the portion to be reoriented if and only if all arcs below the dropped arc have positive reverse leeway after the flow change, which is equivalent to requiring that the outgoing arc be the lowest blocking arc.

Network Alternating Path Algorithm

On the basis of the foregoing discussion, the rules of the NAP algorithm can be stated as follows:

(1) Apply the initialization procedures of Theorem 1 to create a feasible starting NAP basis for the capacitated transshipment problem (possibly containing artificial arcs). Determine node potentials such that the reduced cost of each basic arc is zero.

(2) Select any profitable nonbasic arc to enter the basis. If no profitable arcs exist, the problem is solved.

(3) Determine the arc to leave the basis using the rules of Theorem 3.

(4) Execute a basis exchange step, reorienting the basis tree so that the root node remains fixed and updating flow values and node potentials. Then return to Step 2.

The special implications of the NAP basis structure for degeneracy are now demonstrated, establishing the finiteness of a primal simplex algorithm restricted to NAP basis structures. The following lemma, in conjunction with the preceding results, provides the key relationship on which this property is based.

Lemma 2: Every pivot that preserves the NAP basis structure (in accordance with the rules of Theorem 3), while maintaining the root node potential constant (e.g., by the standard updating procedure of [11]), has exactly one of three effects on the remaining node potentials of the network:

i) The outgoing arc is the incoming arc. In this case no node potentials change.

ii) If the outgoing arc lies on the z-q path, then the node potentials for a non-empty subset of the nodes strictly increase and the remaining node potentials are unchanged.

iii) If the outgoing arc lies on the z-p path, then the node potentials for a non-empty subset of the nodes strictly decrease and the remaining node potentials are unchanged.

Proof: Recall from the convention concerning the specification of the nodes p and q, that the incoming arc is identified as (p,q) with reduced cost $c_{pq} + \pi_p - \pi_q < 0$ if the arc has zero flow and is identified as (q,p) with reduced cost $c_{qp} + \pi_q - \pi_p > 0$ if the arc has saturating flow. Case i is obvious. Next suppose the outgoing arc lies on the z-q path. The standard updating rule of [11] assigns the same change in node potential to every node of the subtree rooted at p (after reorienting the path from the dropped arc to p). This node potential change is precisely $\Delta = (c_{qp} + \pi_q - \pi_p)$ if (q,p) is the incoming arc and $\Delta = -(c_{pq} + \pi_p - \pi_q)$ if (p,q) is the incoming arc. In each case, $\Delta > 0$. Finally, if the incoming arc is on the z-p path, then the standard updating rule of [11] assigns a potential change to each node of the subtree rooted at q, which is the negative of the change previously indicated for each case. This completes the proof.

Theorem 4: The NAP basis algorithm is finite and independent of the choice of incoming arc.

Proof: It suffices to show that the number of degenerate pivots that can occur in unbroken succession is finite since the method can only make a finite number of nondegenerate pivots. By Lemma 2, the node potentials are changing in a uniform direction throughout a sequence of degenerate pivots and at least one node potential changes on each pivot. Since the root node maintains a constant potential and the values of the other potentials are thus uniquely determined for any basis, it follows that no basis can repeat during this succession of pivots, completing the proof.

5. COMPUTATIONAL CONSIDERATIONS

The rules of the NAP algorithm identify the arc to leave the basis as either the highest or lowest qualifying arc on a particular segment of the basis equivalent path. The specialized labeling procedures developed for implementing primal simplex network algorithms as described in Section 2 can be used to efficiently identify these path segments and the outgoing arc; thus, it is possible to incorporate previously proven means for accelerating the solution process into the NAP algorithm.

In the case of a degenerate pivot, the arc to leave the basis is the lowest arc that qualifies on the specified path segment (i.e., the z-p path). Thus, as soon as degeneracy is detected on this path, a pivot can be performed. By using the special labeling functions to identify this path quickly, degeneracy should be detected earlier, thus requiring less processing time to perform degenerate pivots. In the case of a nondegenerate pivot, no disadvantage is incurred by dropping the highest qualifying arc since it is necessary to conduct a full trace of the arcs in the basis equivalent path in any event.

The unique form of the NAP algorithm should provide for computational efficiencies over previous codes by

(1) using previously developed implementation schemes for minimizing the calculations at each iteration and executing degenerate pivots relatively faster,

(2) explicitly bypassing all non-NAP bases so as to reduce the percentage of degenerate pivots.

In order to test the validity of the above statements, it was necessary to develop a computer code for the NAP algorithm and compare it against other efficient codes for solving network problems.

The fastest primal simplex capacitated transshipment code currently available is ARC-II [2]. This code is written in FORTRAN, using a modular design with several subroutines. The code uses the predecessor, thread, and cardinality functions discussed in Section 2 and another function called the last node function [2] to maintain and update the basis data.

ARC-II was modified to incorporate the NAP algorithm by changing the outgoing arc decision rule and substituting an artificial NAP-basis start for its usual advanced start. This new code is called ARC-NAP.

To allow comparisons of the algorithms in isolation of other factors, identical pivoting strategies and starting bases were used in the codes tested. The code referred to herein as ARC-II employed the artificial NAP-basis start, a move which degraded the original code's solution times. Also, both programs used the pivot selection procedures described in [9, 10]. Therefore, the solution times reported below can be greatly improved through the use of an advanced start and candidate list type pivot selection rules.

The codes were tested on a common set of randomly-generated test problems [13] which included assignment and transshipment problems. Problem specifications are given in Table III. Problems 1 through 5 are 400-node assignment problems, problems 6 through 19 are 400-node transshipment problems, and problems 20 through

23 are 1000-node transshipment problems. All problems were solved on a UNIVAC 1108 using the FORTRAN-V version 11A compiler. The computer jobs were executed during periods when the machine load was approximately the same, and all times are exclusive of input and output. The total time spent solving the problem was recorded by calling a CPU clock upon starting to solve the problem and again when the solution was obtained.

The solution times for each code are contained in Table IV. The results show ARC-NAP to be uniformly superior to ARC-II on the assignment problems and 1000-node transshipment problems. Based on the sum of solution times for these problems, ARC-NAP is roughly 21% and 10% faster, respectively, on these problems than ARC-II. The times for the 400-node transshipment problems are approximately equal for the two codes.

Table III
Problem Specifications

	PROBLEM NO.	TOTAL NUMBER NODES	SOURCE NODES	SINK NODES	TOTAL NUMBER ARCS
Assignment Problems	1	400	200	200	1500
	2	400	200	200	2250
	3	400	200	200	3000
	4	400	200	200	3750
	5	400	200	200	4500
Transshipment Problems	6	400	8	60	1306
	7	400	8	60	2443
	8	400	8	60	1306
	9	400	8	60	2443
	10	400	8	60	1416
	11	400	8	60	2836
	12	400	8	60	1416
	13	400	8	60	2836
	14	400	8	60	1382
	15	400	8	60	2676
	16	400	8	60	1382
	17	400	8	60	2676
	18	400	8	60	1306
	19	400	8	60	2443
	20	1000	50	50	2900
	21	1000	50	50	3400
	22	1000	50	50	4400
	23	1000	50	50	4800

Table IV

Solution Times in Seconds
on UNIVAC 1108

PROBLEM TYPE	PROBLEM NO.	TOTAL TIME		START TIME		SOLUTION TIME	
		ARC-II	ARC-NAP	ARC-II	ARC-NAP	ARC-II	ARC-NAP
Assignment	1	2.993	2.032	.036	.027	2.897	2.005
	2	3.575	2.685	.031	.032	3.545	2.653
	3	3.264	3.113	.030	.032	3.234	3.081
	4	3.967	3.266	.036	.032	3.931	3.234
	5	4.563	3.398	.037	.032	4.526	3.365
Sum of Total Times		18.362	14.494				
400-Node Transship-ment	6	2.226	2.478	.030	.029	2.196	2.449
	7	2.395	2.819	.031	.039	2.364	2.780
	8	1.872	2.177	.035	.033	1.837	2.144
	9	2.822	3.141	.035	.047	2.787	3.094
	10	2.062	2.195	.029	.037	2.034	2.158
	11	3.971	3.812	.032	.036	3.939	3.776
	12	2.129	2.089	.033	.035	2.097	2.054
	13	2.996	2.807	.027	.032	2.969	2.775
	14	2.536	2.506	.032	.035	2.496	2.471
	15	3.317	2.958	.032	.037	3.285	2.922
	16	2.462	2.279	.033	.031	2.429	2.247
	17	2.654	2.193	.034	.028	2.620	2.165
	18	2.122	2.574	.034	.038	2.088	2.536
	19	2.791	3.159	.034	.040	2.757	3.119
Sum of Total Times		36.355	37.187				
1000-Node Transship-ment	20	9.718	9.808	.074	.076	9.643	9.732
	21	10.936	10.221	.075	.076	10.861	10.145
	22	12.236	11.644	.075	.074	12.160	11.570
	23	15.196	11.962	.101	.076	15.095	11.886
Sum of Total Times		48.086	43.635				

In order to pinpoint the areas in which computational efficiencies were achieved, several problem statistics were collected. These included total number of pivots and total number of degenerate pivots. In addition, for the 1000-node networks, average time for performing a nondegenerate pivot and average size of the subtree updated at each iteration were calculated. These statistics are contained in Table V.

These statistics contradict the original expectation that the algorithm would reduce the percentage of degenerate pivots. The percentage of degenerate pivots increases uniformly for all problems. In terms of total pivots performed, there is a decrease for the assignment problems and the 1000-node problems but an increase for the 400-node problems. In addition, the statistics show that the NAP-algorithm requires more time to perform a nondegenerate pivot and updates a larger subtree, on the average, at each iteration than ARC-II.

The computational efficiencies gained by ARC-NAP must, therefore, result from performing fewer total pivots where a larger percentage of these pivots are degenerate combined with the ability to perform degenerate pivots very efficiently. These results suggest that the NAP algorithm is able to quickly scan through a series of alternative basis representations for an extreme point until one is found that allows a relatively "better" nondegenerate pivot to be performed than in the standard simplex network algorithm. Fewer extreme points are examined (and hence fewer nondegenerate pivots performed) before optimality is reached. The algorithm is therefore able to offset the extra computational effort required in performing nondegenerate pivots and updating a larger subtree.

This study indicates that the NAP algorithm is more efficient (or no less efficient) than current state-of-the-art network codes when an artificial start and the pivot rules in [9, 10] are used. Further computational studies are needed to determine why fewer extreme points are examined by the NAP algorithm and

Table V

Problem Statistic

PROBLEM TYPE	PROBLEM NO.	ARC-II TOTAL NO. OF PIVOTS	ARC-II TOTAL NO. OF DEG.* PIVOTS	ARC-NAP TOTAL NO. OF PIVOTS	ARC-NAP TOTAL NO. OF DEG.* PIVOTS	ARC-II TIME PER NON-DEG.* PIVOT	ARC-II AVERAGE NO. OF NODES PER SUBTREE	ARC-NAP TIME PER NON-DEG.* PIVOT	ARC-NAP AVERAGE NO. OF NODES PER SUBTREE
Assign-	1	1739	973	1332	1035				
ment	2	2038	1253	1518	1236				
Problems	3	1732	994	1628	1299				
	4	1681	999	1511	1203				
	5	1838	1107	1578	1274				
TOTALS		9028	5326	7567	6047				
Pivots - % Deg.*			59		80				
400-Node	*6*	1543	1200	1568	1279				
Transship-	7	1663	1333	1852	1554				
ment	8	1372	1098	1546	1278				
Problems	9	1801	1402	1755	1481				
	10	1360	999	1427	1123				
	11	2337	1570	2046	1584				
	12	1495	1079	1410	1128				
	13	1902	1371	1781	1298				
	14	1550	1098	1643	1278				
	15	1971	1456	2010	1636				
	16	1560	1102	1598	1270				
	17	1534	1179	1453	1194				
	18	1372	1098	1546	1278				
	19	1801	1402	1755	1481				
TOTALS		23261	17387	23390	18962				
Pivots - % Deg.*			75		81				
1000-Node	20	3468	2717	3537	3000	0.002	26.6	0.003	23.1
Transship-	21	3941	3097	3651	3071	0.002	24.4	0.003	26.2
ment	22	4285	3360	4263	3618	0.002	21.8	0.003	25.1
Problems	23	4327	3234	4377	3699	0.002	23.4	0.003	25.9
TOTALS		16021	12403	15328	13388		96.2		100.3
Pivots - % Deg.*			77		85				

*Degenerate

the effects of various start procedures and pivot strategies on the performance of the NAP algorithm. Refinements to the ARC-NAP code based on the insights gained from such studies promise even further improvements in solution times.

REFERENCES

1. R. Barr, F. Glover, and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," *Mathematical Programming* 7, 1 (1974), 60-89.

2. R. Barr, F. Glover, and D. Klingman, "Enhancements to Spanning Tree Labeling Procedures for Network Optimization," Research Report CCS 262, Center for Cybernetic Studies, The University of Texas at Austin (1976).

3. R. Barr, F. Glover, and D. Klingman, "A New Alternating Basis Algorithm for Semi-Assignment Networks," Research Report CCS 264, Center for Cybernetic Studies, The University of Texas at Austin (1977).

4. R. Barr, F. Glover, and D. Klingman, "The Alternating Basis Algorithm for Assignment Problems," *Mathematical Programming* 13 (1977), 1-13.

5. R. Barr, F. Glover, and D. Klingman, "The Generalized Alternating Path Algorithm for Transportation Problems," Research Report CCS 282, Center for Cybernetic Studies, The University of Texas at Austin (1977), to appear in *European Journal of Operations Research*.

6. W. Cunningham, "A Network Simplex Method," *Mathematical Programming* 11 (1976), 105-116.

7. J. Edmonds, "Paths, Trees and Flowers," *Canadian Journal of Mathematics* 77 (1965), 449-467.

8. J. Edmonds, "Maximum Matching and the Polyhedron with 0-1 Vertices," *Journal of Research of the National Bureau of Standards* 69B (1965), 125-130.

9. F. Glover, D. Karney, and D. Klingman, "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," *Networks* 4, 3 (1974), 191-212.

10. F. Glover, D. Karney, D. Klingman, and A. Napier, "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," *Management Science* 20, 5 (1974), 793-813.

11. F. Glover, D. Klingman, and J. Stutz, "Augmented Threaded Index Method for Network Optimization," *INFOR* 12, 3 (1974), 293-298.

12. D. Karney and D. Klingman, "Implementation and Computational Study on an In-Core Out-of-Core Primal Network Code," *Operations Research* 24 (1976), 1056-1077.

13. D. Klingman, A. Napier, and J. Stutz, "NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," *Management Science* 20, 5 (1974), 814-821.

14. V. Srinivasan and G. Thompson, "Accelerated Algorithms for Labeling and Relabeling of Trees with Applications for Distribution Problems," *JACM* 19, 4 (1972), 712-726.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Center for Cybernetic Studies The University of Texas	Unclassified
	2b. GROUP

3. REPORT TITLE

A Network Augmenting Path Basis Algorithm for Transshipment Problems

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

Richard Barr — Joyce Elam
Fred Glover — Darwin Klingman

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August 1977	32	14

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-76-C-0383; 75-C-0616; 0569 b. PROJECT NO. NR047-021 c. d.	Center for Cybernetic Studies Research Report CCS 272 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Office of Naval Research (Code 434) Washington, D.C.

13. ABSTRACT

The purpose of this paper is to present a new simplex algorithm for solving capacitated transshipment network problems which both circumvents and exploits the pervasive degeneracy in such problems. This generalized alternating path algorithm is based on the characterization of a special subset of the bases that are capable of leading to an optimal solution. With consideration restricted to these bases, fewer alternative representations of a given extreme point are inspected. The impact on the number of degenerate pivots and problem solution times is demonstrated by computational testing and comparison with other approaches.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Network						
Transshipment						
Degeneracy						
Transportation						
Optimization						
Linear Programming						